Data Mining Project

Aprendizagem Computacional (AC) — 2021/22

Grupo 26

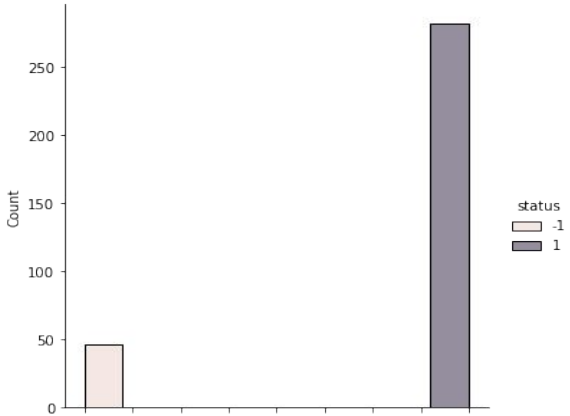
Ana Barros – up201806593@edu.fe.up.pt
João Costa – up201806560@edu.fe.up.pt
João Martins – up201806436@edu.fe.up.pt

Business understanding

- Businesses exist to make money, and loans are the way a bank makes money. The bank is interested in loaning, so it gains wealth, but not every loan will generate wealth. These loans need to be rejected.
- For a bank, it is better to reject loans that should have been accepted instead of accepting loans which should have been rejected.
- From this analysis, it is safe to conclude that the bank has two main goals: minimize money loss, and maximize profit. This can be translated to the following data mining goal: **predict which loans to reject**.
- When analyzing the data, it is important to only consider information that was available at the time of the decision. For example, only cards issued before the loan request and transactions/balances that refer to a point in time before the request are to be considered.

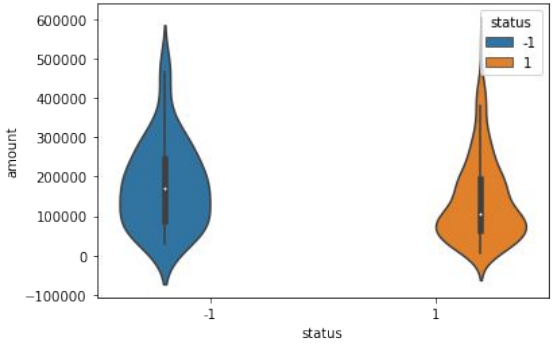
DU: Domain Analysis

Almost 90% of the loans in the dataset have been granted. This means that accuracy isn't a good metric to optimize for.

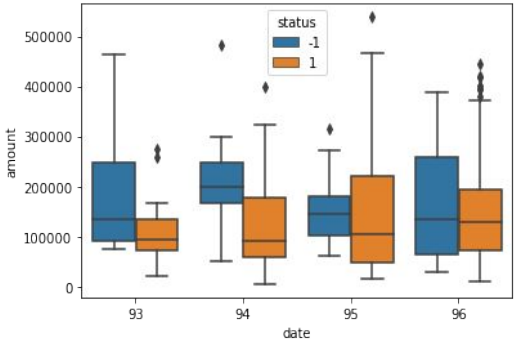


Loan status distribution

Since the status doesn't form a normal distribution, we used the **Spearman correlation** to filter features. This is paired with the ANOVA test.



Loan amount by status

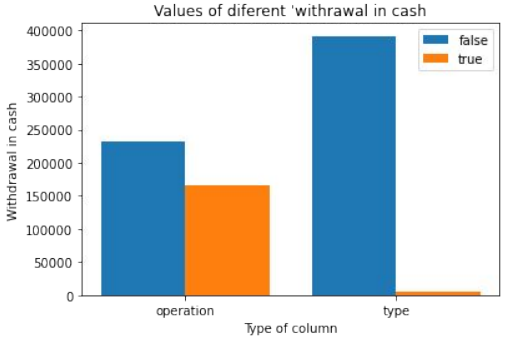


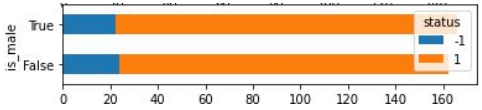
Loan status by amount and date

- There appears to be less rejected loans at higher amounts. This can be misleading since it is impossible to know if the currencies in the dataset are in constant prices. They should be, so the **effects of inflation are disregarded**.
- The yearly temporal analysis doesn't yield any meaningful results.
- Since the data is old, almost no clients in the dataset had a card: only 11 clients with a card have ever requested a loan.
- The median monthly income in the region at the time was 3200CZK.

DP: 6 Dimensions of data quality

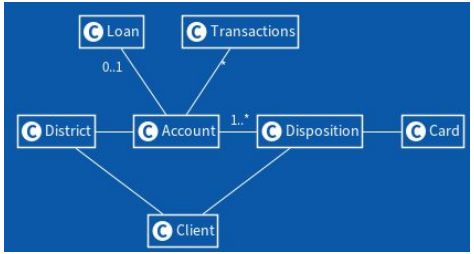
1. **Completeness**
   There are two non-optional values missing in the district data. Meanwhile, in the *Transaction* data, there were attributes with a large quantity of null values (≈80%). Since some non-optional data was missing, the data **cannot be considered complete**.
2. **Consistency**
   The overlapping information in the *type* and *operation* attributes has contradictions. This means that the data is **not consistent**.
3. **Conformity**
   Dates follow the format ***yymmdd***, except in the case of the client's *birthdate* attribute. This attribute encoded the genre information as well, by adding 50 to the month (*mm*). This means that the data **lacks conformity**.
4. **Accuracy**
   According to this research paper, there is a bias regarding the sex of the borrower and loan officers, which is not present on the dataset. For this reason, we assume that our data is **not accurate**.
5. **Integrity**
   Most accounts have no loan requests associated with them: there are orphaned records. This mean that the data **lacks integrity**.
6. **Timeliness**
   The dataset is at least 20 years old. This means that the information contained in it is quite old. As such, it **doesn't achieve timeliness**.

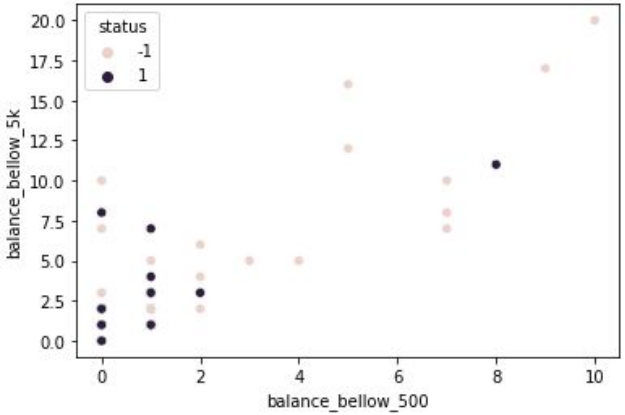




DP: Data transformation — join data

- The tables composing the dataset needed to be joined into a single table: table keys aren't part of the final dataset.
- No duplicated entries. Some tables have relations that aren't one-to-one: *Transactions*, *Disposition*, *Card*.
- The only data considered is the data referring to before the loan request.
- The account's location (district) was ignored.



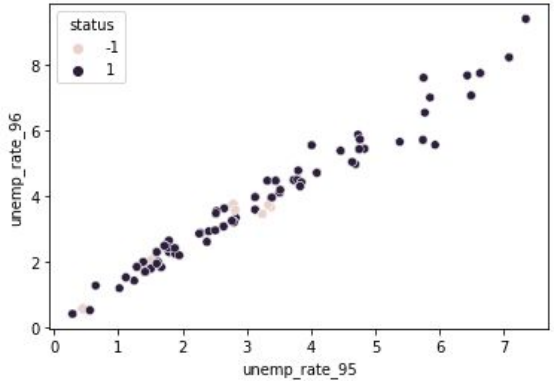


- There is an attribute depicting the type card associated with each account, in order to condense the card information: no card, junior, classic, and gold.
- The main attributes extracted from the *Transaction* table: the number of times the client was sanctioned for negative balance; mean household payments amount; mean interest payments amount; mean balances on the last year, semester, and month; the number of time the account balance was above 500 and below 5000.
- Since an account can be co-owned, a new boolean attribute reflects whether the account has a *disponent owner* or not.

DP: Data transformation — missing values and categories

Many of the algorithms used didn't accept missing values and/or categorical attributes, so these had to be dealt with.

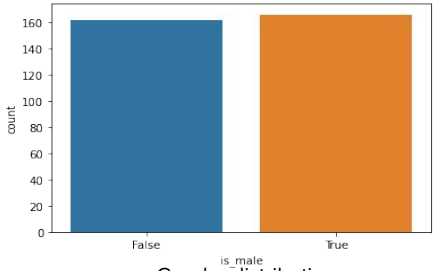
- The MICE multivariate inputer filled the missing values on the district data: unemployment rate and crime count in '95. This works well, because there is a (almost) linear relation between these attribute and their 96' counter-parts.
- The card column created is categorical. Since these values have a hierarchy, they can be translated to integers: **0** - no card; **1** - junior; **2** - classic; **3** - gold.
- The issuance frequency based on its temporal properties. It was considered that people only do 1 transaction a day. As such: **1** - issuance after transaction; **7** - weekly issuance; **30** - monthly issuance.
- The district name attribute has a high cardinality and no implicit order. The **CatBoost** encoder was used, as an alternative to **Leave One Out** encoding, to encode this attribute. The district code was dropped, because it was implying an order that didn't exist.
- Since most of the data referring to the target bank/account (≈83%) of the transactions was missing, these attributes were dropped.



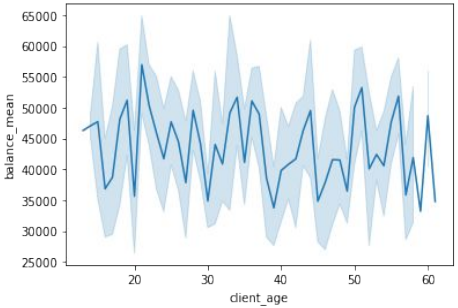

```
interest credited                          70761
payment for statement                      58377
household                                  42839
                                           19065
old-age pension                            13502
insurrance payment                          6592
sanction interest if negative balance        305
Name: k_symbol, dtype: int64
```

Feature engineering — client

The clients' gender and age extracted from the birth number.

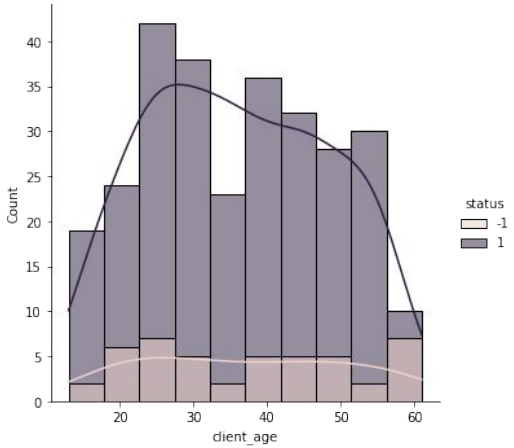


Gender distribution

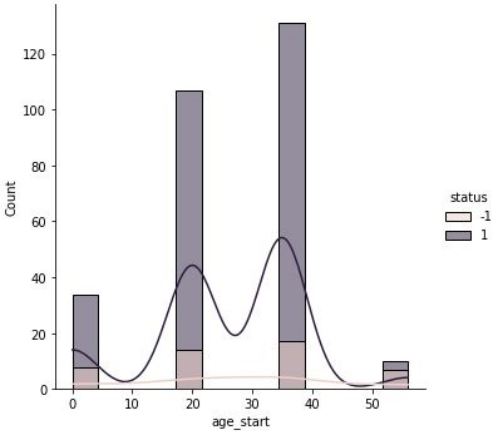


Age distribution by mean balance

The data shows that older clients (above the retirement age) tend to have more rejected loans. The wealth appears to be evenly distributed between all age groups.



Age distribution

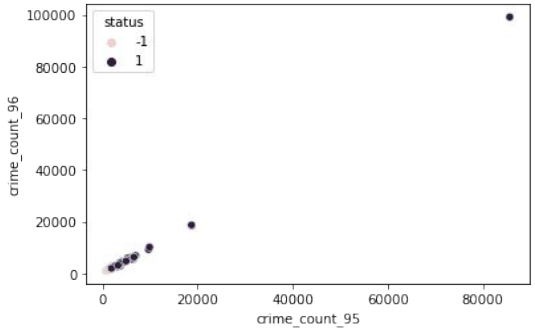


Discretized age using bucket start age

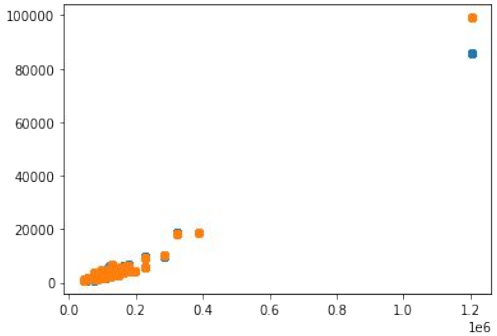
Feature engineering — district

Both the unemployment rate and crime counts in 1995 and 1996 are almost linearly correlated:

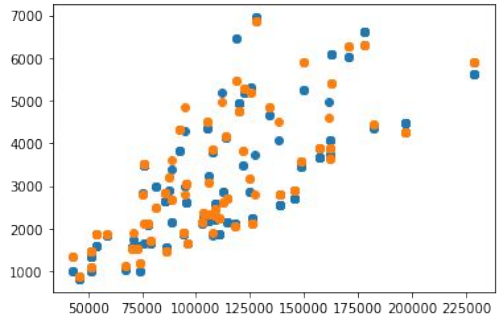
- The 2 unemployment rate attributes were substituted by an unemployment growth attribute.
- There are 3 regions with a much higher number of committed crimes: Even in the case of these regions, the relation is almost linear. Also, as expected, regions with higher population have more crimes committed and this remains true for both years.
- The crime counts were also correlated with the population size. To solve this, the two attributes became *crime per capita* and were merged into a *crime per capita growth*.



Relation between *crime_count*s (95 & 96) attributes



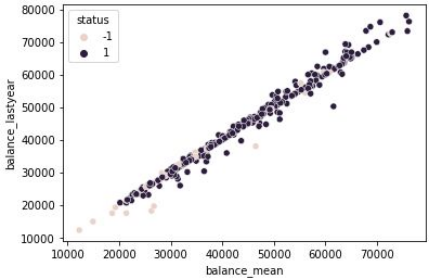
Relation between population size and *crime_count*s (95 & 96) attributes

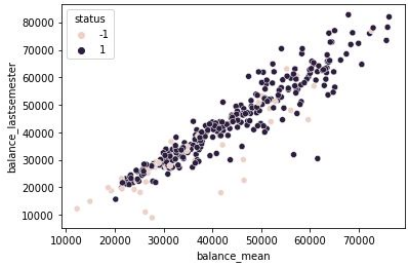


Relation between population size and *crime_count*s (95 & 96) attributes (zoomed in)

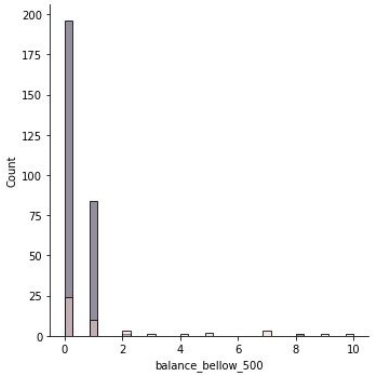
Feature engineering — transactions

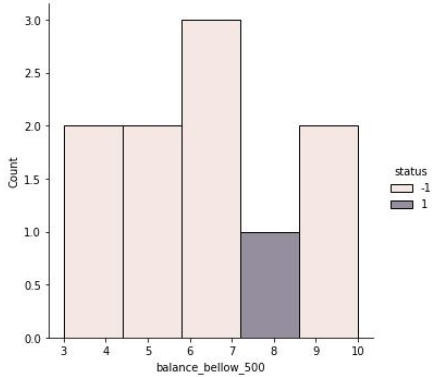
There is a high correlation between the mean balances of the accounts globally, on the last year, on the last semester, and on the last month.



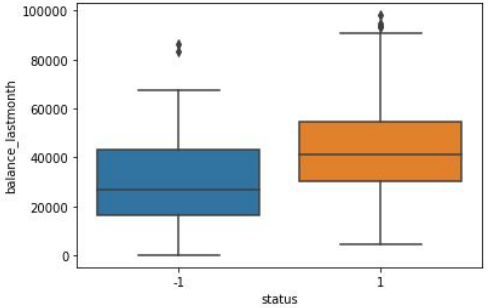


To solve this, the three features were replaced by their perceptual difference from the account's mean balance.



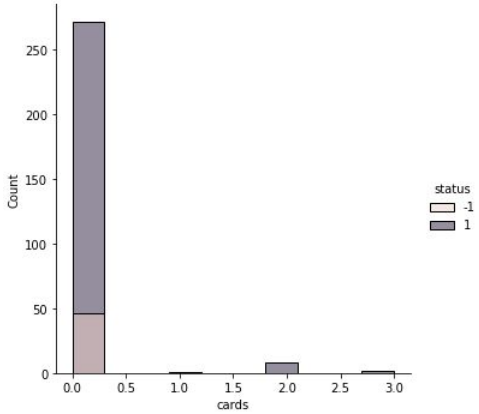


There is an outlier when examining the number of times the account had a balance below 500.



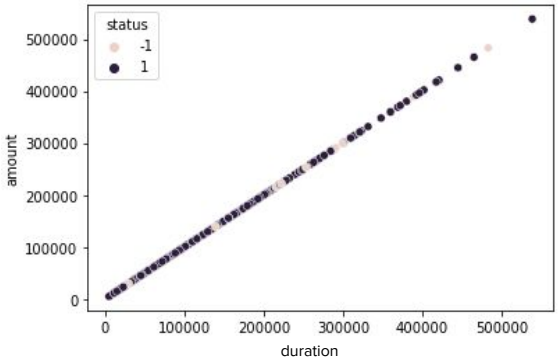
Feature engineering — others

- All dates were converted to ages relatively to loan request date, e.g., client age, account age, etc...
- The card attribute is biased: almost no cards in the dataset (11) and all accounts with a card have been granted a loan. This can be misleading, so the attribute was dropped.



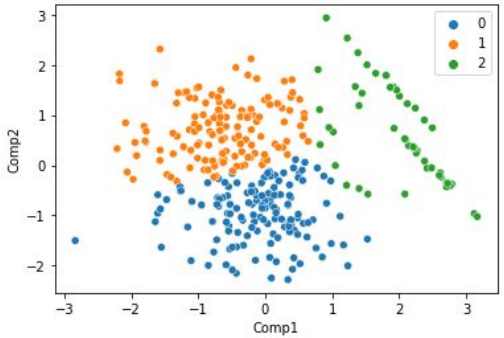
- There is a linear relation between the amount of a loan, and the payments times the duration.
- The duration has low correlation with the label and scores badly on the ANOVA test.
- The amount scores lower on the ANOVA test than the payments, so only the payments attribute was kept.

	Features	Score
0	payments	9.518871
2	amount	5.450596
1	duration	0.004079

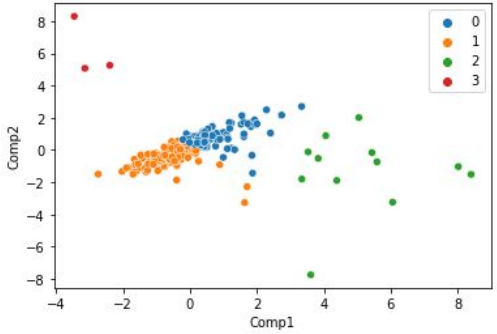


Clustering

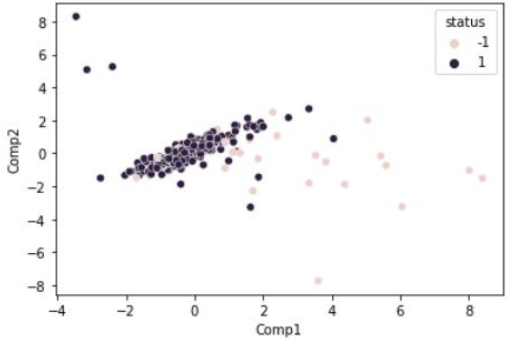
- **PCA** was used to reduce the dimensionality of our features.
- **K-Means** was used to generate the clusters because it is a fast algorithm, and it uses a stochastic approach that frequently works well.
- To find the optimal number of clusters, the **elbow method** was used, where we calculated the distortion and chose the value **k** by analyzing the plot.
- For clients' clusters, **3 clusters were formed**, with the green one being the most clear.
- Regarding transactions' clusters, the red, and the green clusters are differentiated from the main group (orange and blue). All loans in the green cluster were accepted, and most loans in the red cluster weren't.



Client's clusters



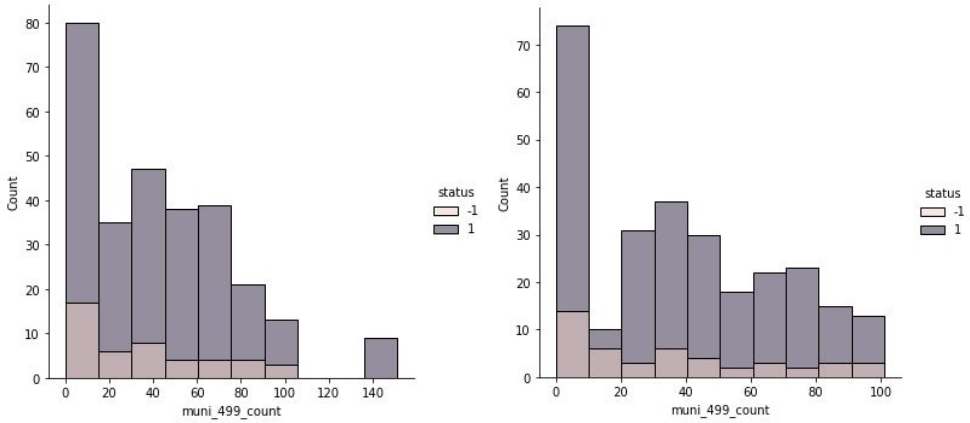
Transaction's clusters

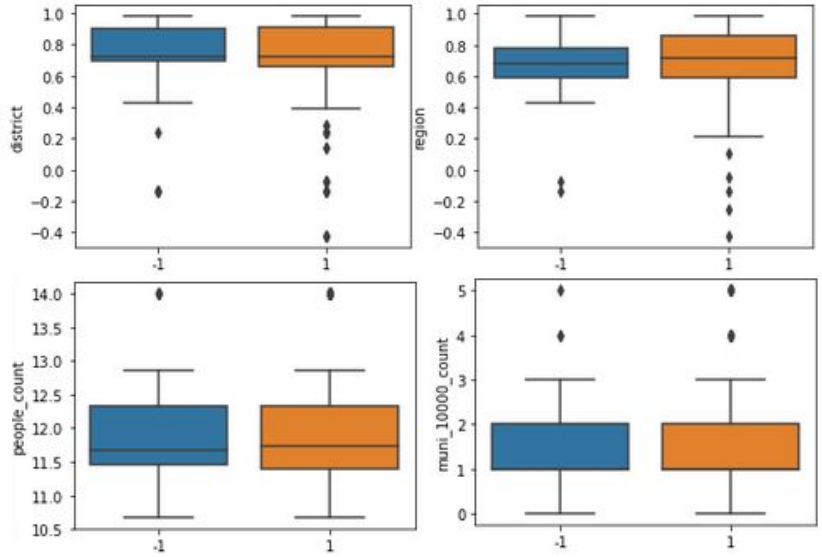


Transaction's clusters with status

Outliers

- The **DBSCAN** algorithm detected that **27%** of our data **were outliers**.
- Using diverse plots we found that the *district*, *region, and municipalities* related features had a significant number of outliers.
- We also found that two abnormally high counts of municipalities with less than **500** population had been granted loans. This would probably mislead the model into thinking that these attributes and phenomenon had importance, and, as such, we dropped them.



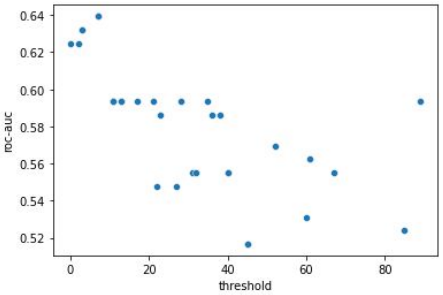


Feature selection

- During feature engineering, attributes were filtered using filter methods: **Spearman correlation** and **ANOVA** test.
- Instead of using a wrapper method for further feature selection, an embedded method was used for its advantages:
  - faster;
  - less prone to overfitting;
  - more accurate.
- The training data was split into a training and a validation set for this task.
- The feature importance reported by the **LightGBM** classifier was used to create subsets of attributes. A threshold was adjusted to compare different subsets of features.
- The best **ROC-AUC** was reported by a threshold of **7**, selecting **28** attributes.
- By selecting a threshold of **38**, a negligible amount of **AUC** is lost (≃ **0.05**), but the number of attributes drops from to **11**.
- By our analysis, the *region* attribute wasn't very relevant. However, the LightGBM model gave some importance to this feature. We believe that the reason for this might be that our encoding might have some leak.

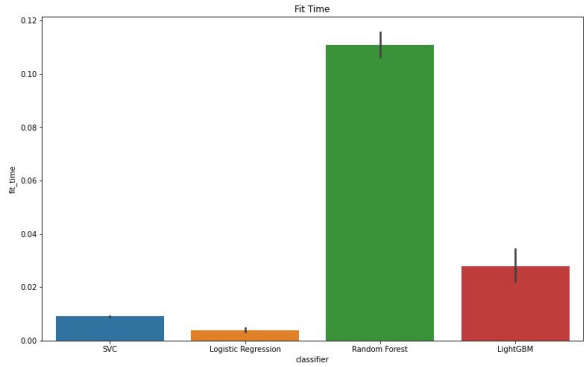
	Features	Score
6	balance_bellow_5k	71.808531
9	balance_growth_lastsemester	21.879819
5	balance_mean	12.747115
1	payments	9.518871
8	balance_growth_lastyear	7.345737
10	balance_growth_lastmonth	7.302655
2	household_mean	0.830395
0	region	0.557291
3	interest_mean	0.387150
7	unem_growth	0.141490
4	balance_max	0.128700

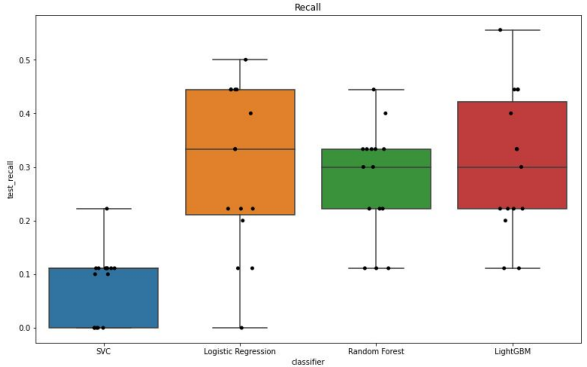
Final set of attributes

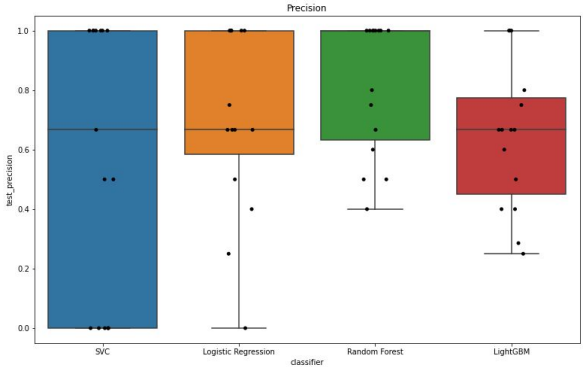


Model baselines

- Rejecting a loan is the positive class.
- Repeated **Stratified K-Fold** with **5 folds** and **3 repeats** is used for cross validation.
- The folds are the same between all models.
- The data is scaled using sklearn's **Standard Scaler** for distance based models, e.g., SVC.
- The Random Forest model is considerably slower than the rest.
- The **SVC is awful** at predicting the positive class.

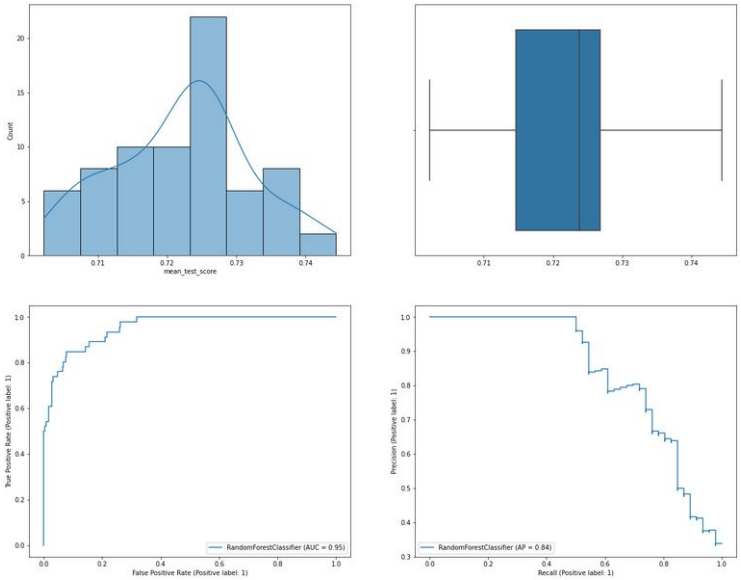


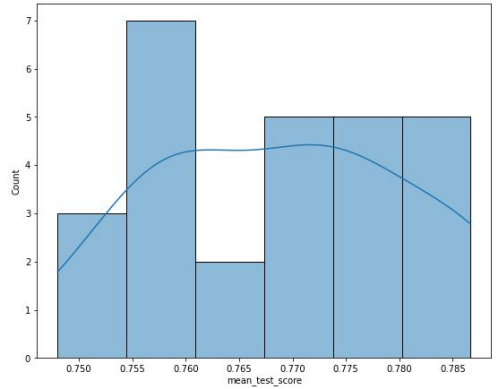


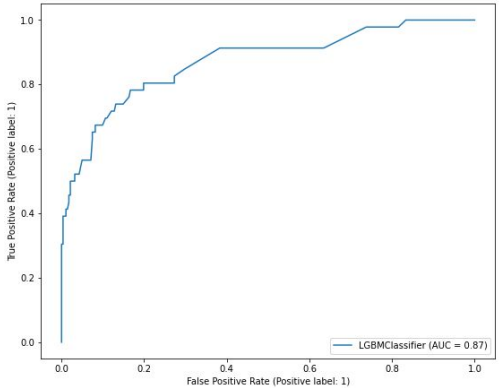


Hyperparameter tuning

- The SVC model improved a lot.
- The Random Forest model is being limited on its max depth and estimators, because it was overfitting on training data.
- The LightGBM model prefers low max depths and number of estimator. This suggests that the model is too complex.

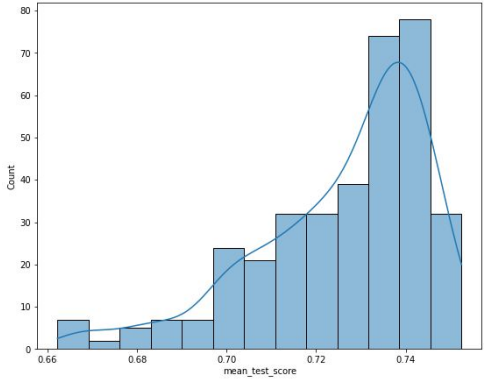


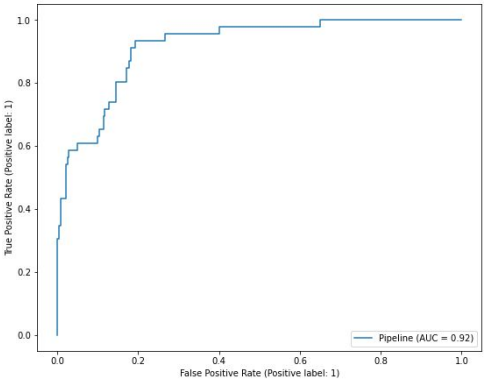




Oversampling

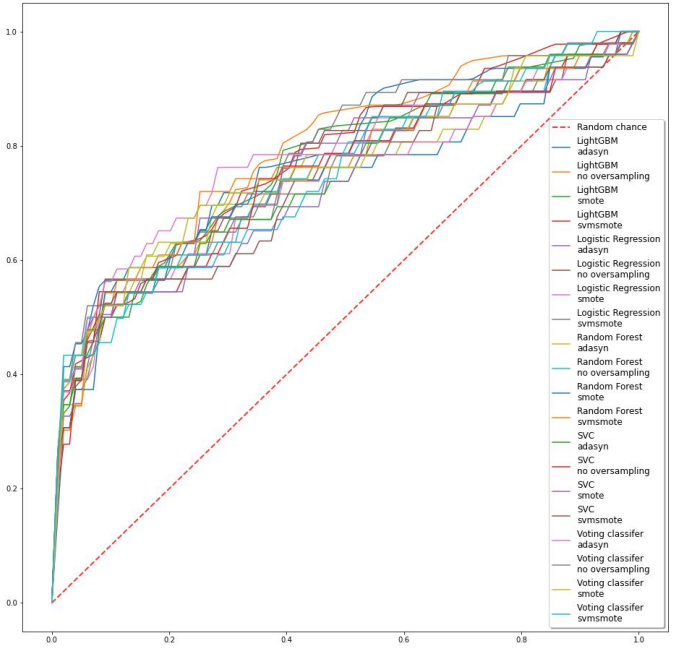
- 3 methods for oversampling were used: **SMOTE**, **SVMSMOTE**, and **ADASYN**.
- **SMOTE:**
  - minority scaled to either 30% or 50% of the majority class;
  - Random Forest stabilized the max depth at 3;
- **SVMSMOTE:**
  - higher sampling strategy in general, e.g., 100% for the Random Forest;
  - LightGBM keeps the lowest sampling strategy;
- **ADASYN:**
  - Low sampling strategies except for the Random Forest (80%);
  - LightGBM saw a change on the parameters (number of estimators), but still seems to consider the model too complex;
  - Obtains the highest ROC-AUC during cross-validation, but the same improvements aren't observed for the test data;





Results

- All models score low in terms of recall: they have trouble finding the loans to reject. The Random Forest model seems to be the best one at this.
- The average ROC curves of all folds of all models are close.
- The LightGBM model remains very stable: results don't vary with/without SMOTE.
- The LightGBM model is reacting badly to the given attributes.
- The Logistic Regression model isn't the worst, but isn't remarkable either.
- By having access to the test data, we'd be able to compare the growth of the train and test errors for the parameters and act accordingly.



Conclusion and future work

- Oversampling techniques didn’t have a positive impact on the test results;
- Having the test data available would help test the models;
- By having more data, the analysis of tendencies could be more robust;
- The test results (Kaggle) were significantly different from the train ones in some cases;
- With more powerful machines, the use of wrapper methods for feature selection should be explored;
- The feature selection stage can be developed further, for example, by extrapolating the unemployment rates for 1994 and 1997, so each loan is associated with the rate of the year it was granted on;
- The models obtained performed well: ≈ 0.96 score on the Kaggle competition.

Report

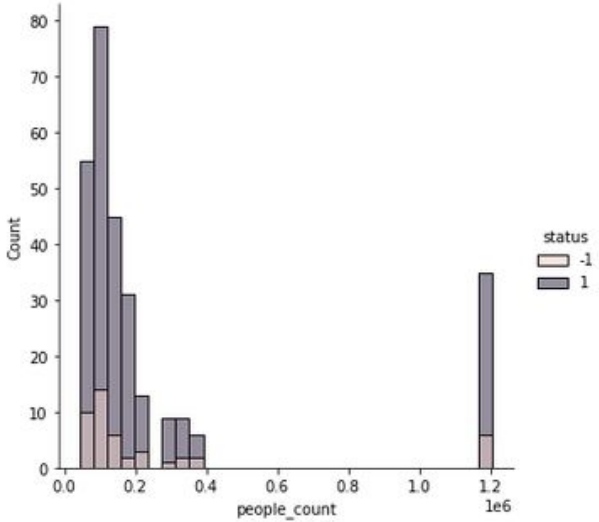
Aprendizagem Computacional (AC) — 2021/22

Grupo 26

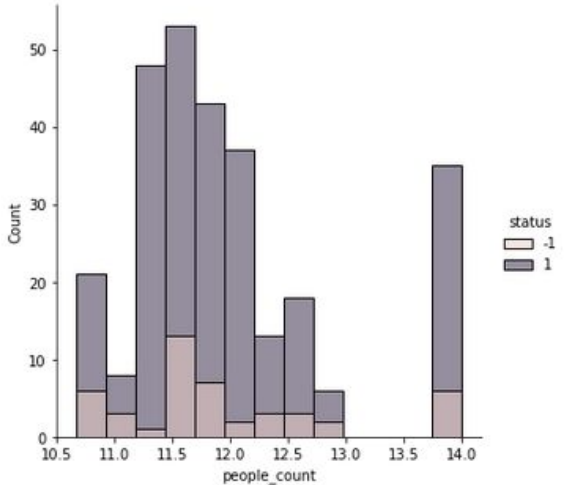
Ana Barros – up201806593@edu.fe.up.pt
João Costa – up201806560@edu.fe.up.pt
João Martins – up201806436@edu.fe.up.pt

Feature Engineering - Skewness

- Some attributes have a skewed distribution
  - for example *avg_salary* of the *District* table.

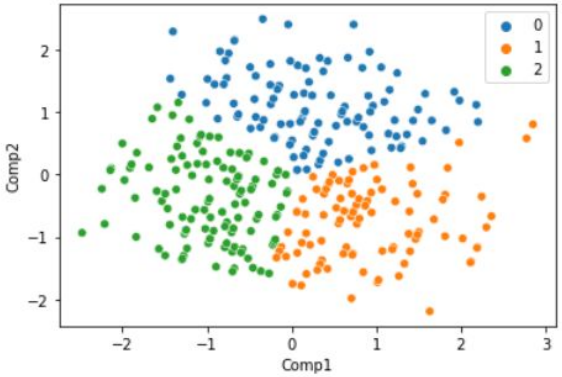


- A logarithm function was applied to these attributes

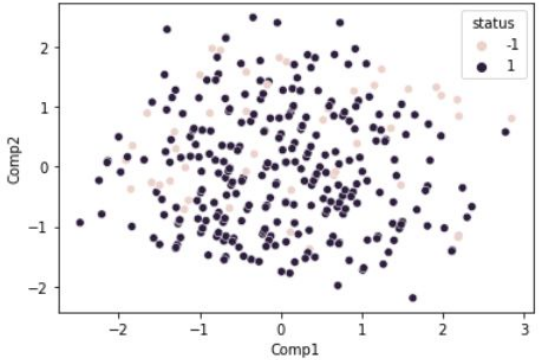


Clustering - Loan

- Clustering was also done with the loan table's attributes, using **PCA** and **KMeans**



Loan's clusters



Loan's clusters with status

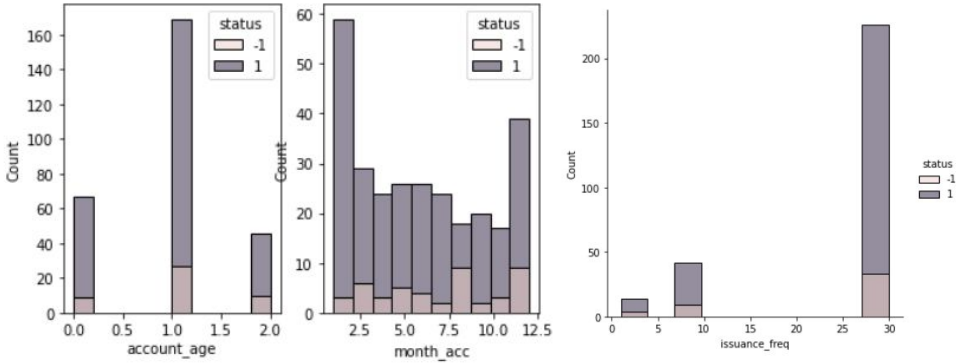
- All loan data seams to be distributed evenly, so no conclusions about loans can be made

Outliers - Account and loan

- Using DBSCAN in a small number of features (3) is not recommended
- Each attribute was analyzed separately, for each table

Account

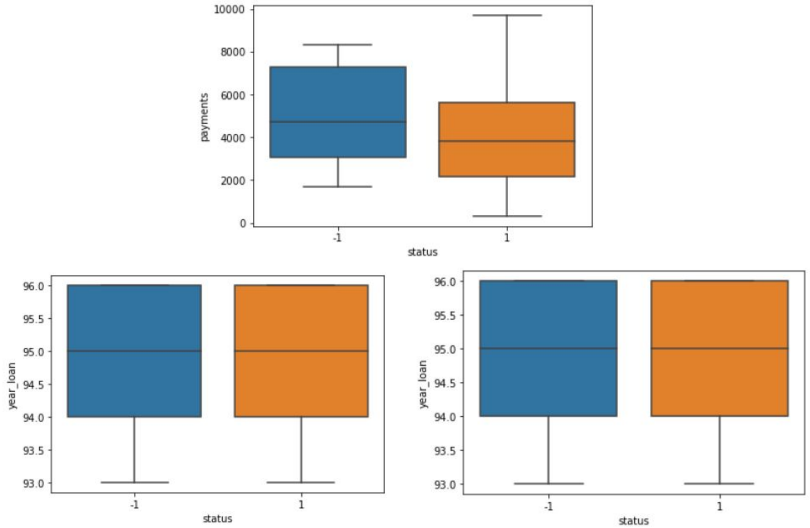
- No outliers found



Account's attributes plots

Loan

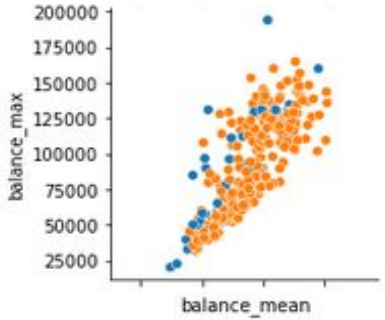
- No outliers found

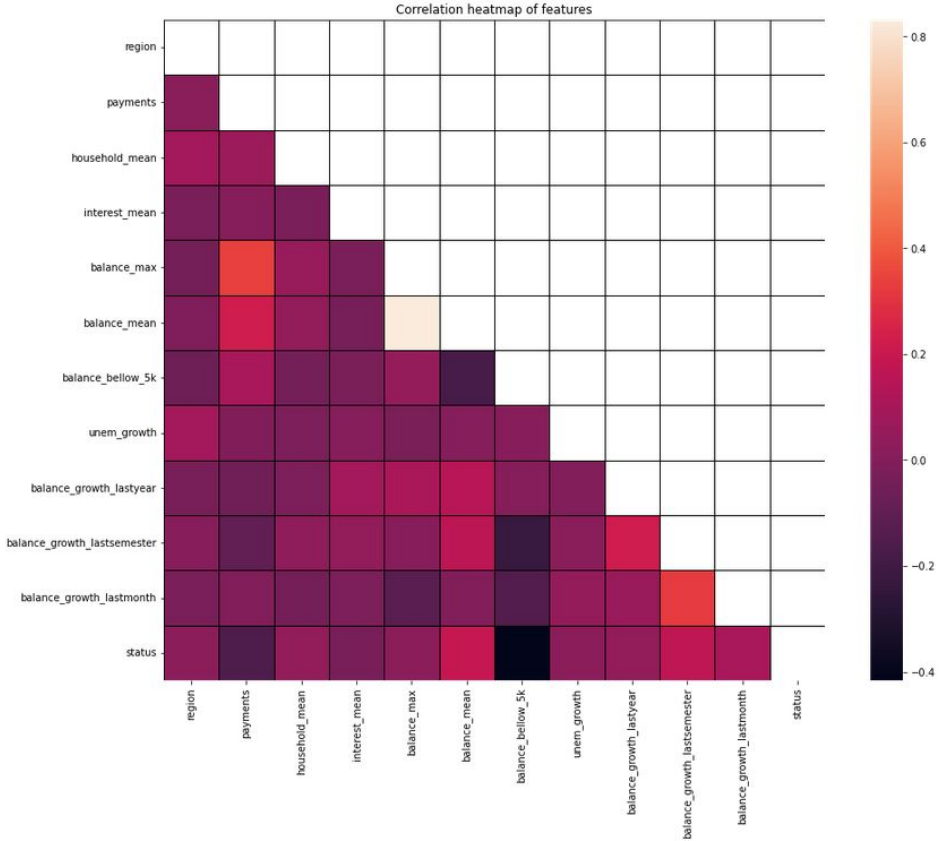


Loan's attributes plots

Feature Selection Results

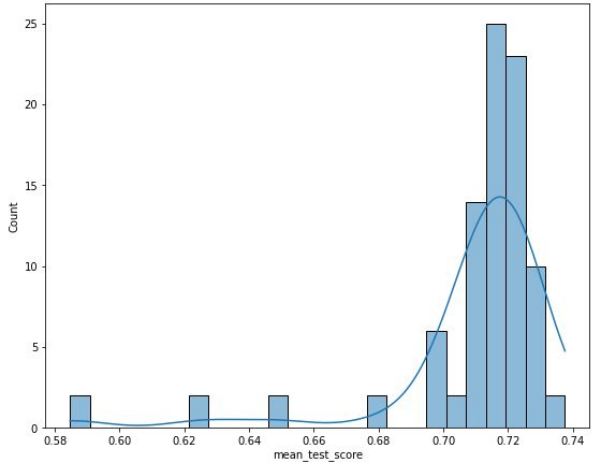
- The selected attributes from the embedded method seem to not be much correlated between each other
- The *balance_max* and *balance_min* attributes are highly correlated, as expected

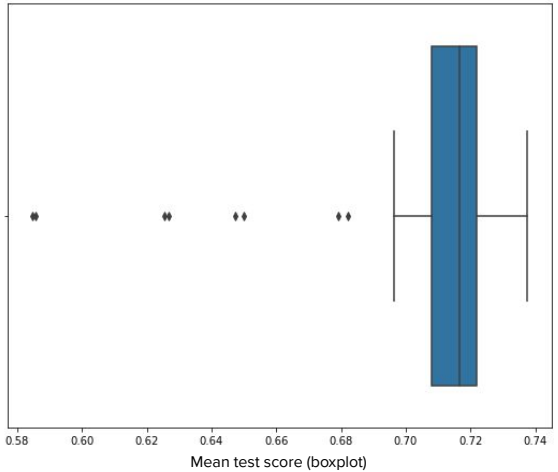


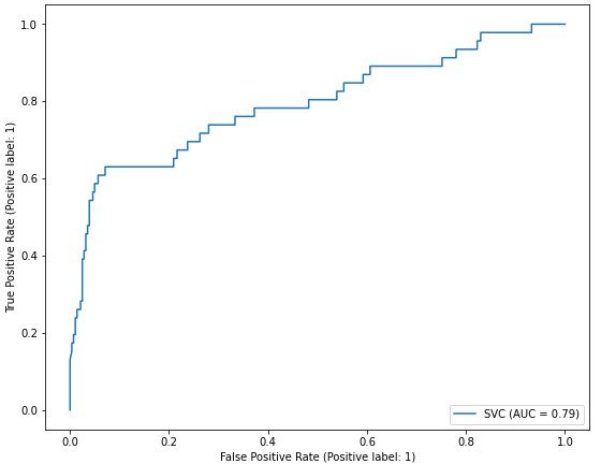


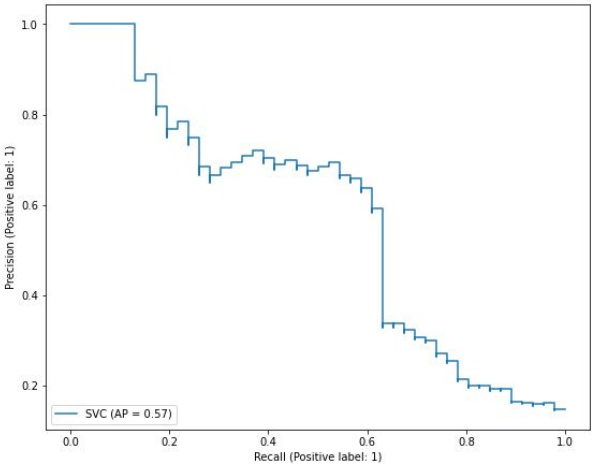
Hyperparameter tuning: Support Vector Classifier

- Parameters:
    - **degree**: The degree of the polynomial used to find the hyperplane to split the data;
    - **C**: Controls the trade-off between smooth decision boundary and classifying the training points correctly;
    - **gamma**: Kernel coefficient for ‘rbf’, ‘poly’ and ‘sigmoid’;
    - **kernel**: Specifies the kernel type to be used in the algorithm.
- Best parameters: {**C**: *0.001*, **degree**: *3*, **gamma**: *scale*, **kernel**: *poly*}
- Increasing **C** leads to overfitting as the classifier tries to perfectly fit the training data.
- SVC improved a lot.


Count
mean_test_score

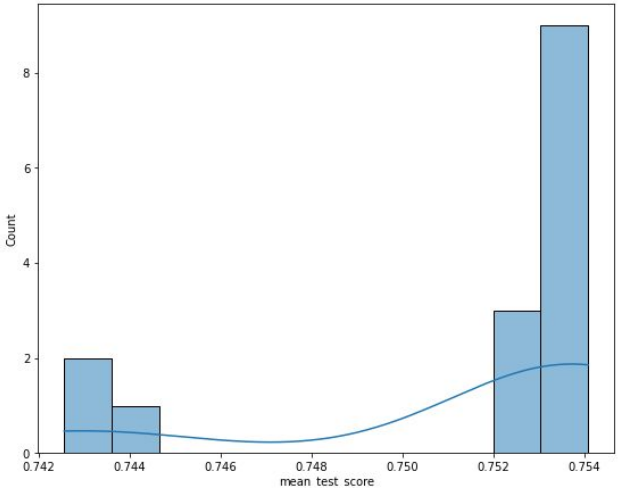

Mean test score (boxplot)

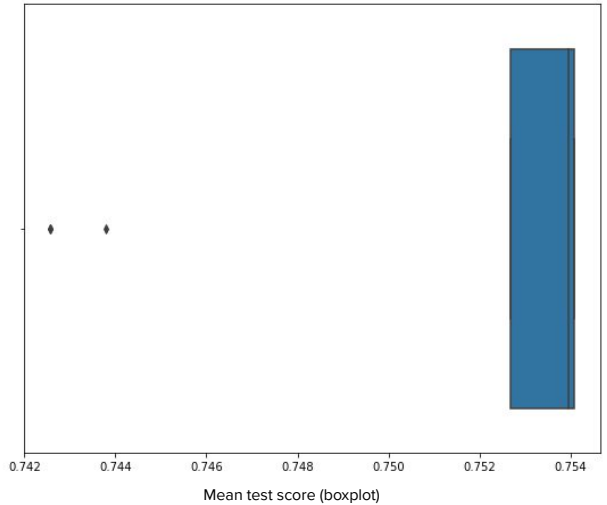

True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
SVC (AUC = 0.79)


Precision (Positive label: 1)
Recall (Positive label: 1)
SVC (AP = 0.57)

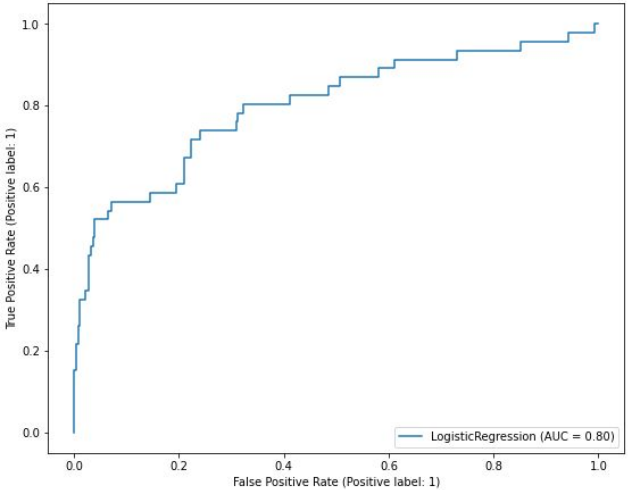
Hyperparameter tuning: Logistic Regression

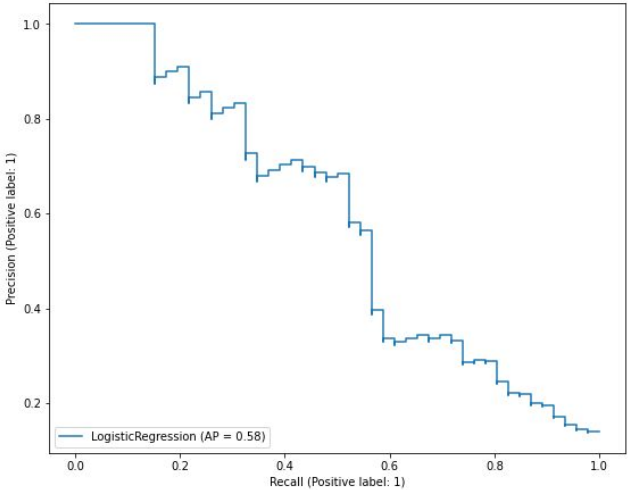
- Parameters:
    - **solver**: Algorithm to use in the optimization problem;
    - **C**: Inverse of regularization strength.
- Best parameters: {**C**: *10*, **solver**: *newton-cg*}
- Even after hyperparameter tuning, we still got bad results.
- Logistic Regression doesn't give us good results even after the grid search, since our data is not linearly separable (like most real world scenarios).
- It wasn't necessary to regulate the **C** value of the model: the default value was the one with the best results.





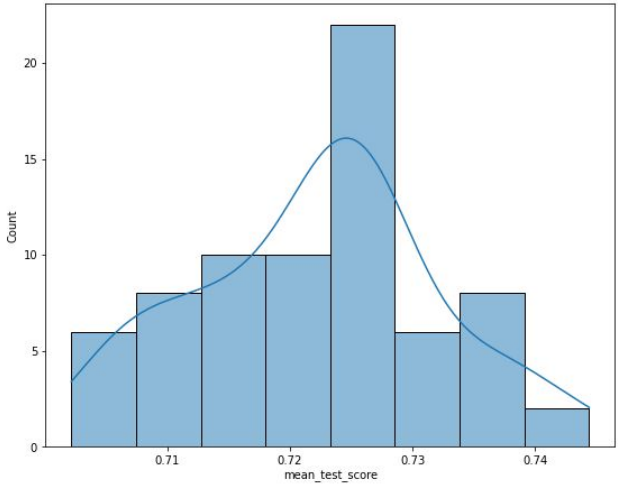
Mean test score (boxplot)

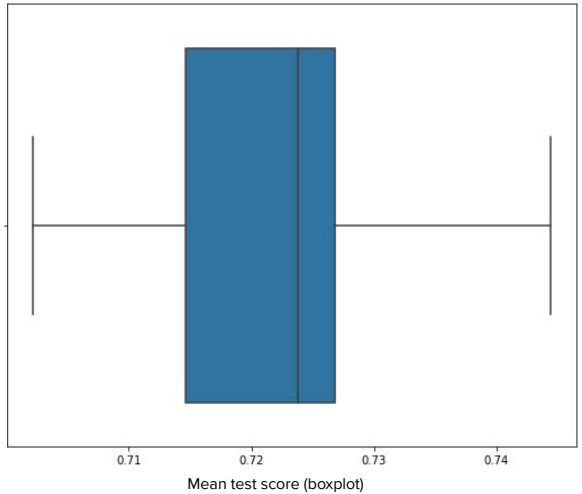


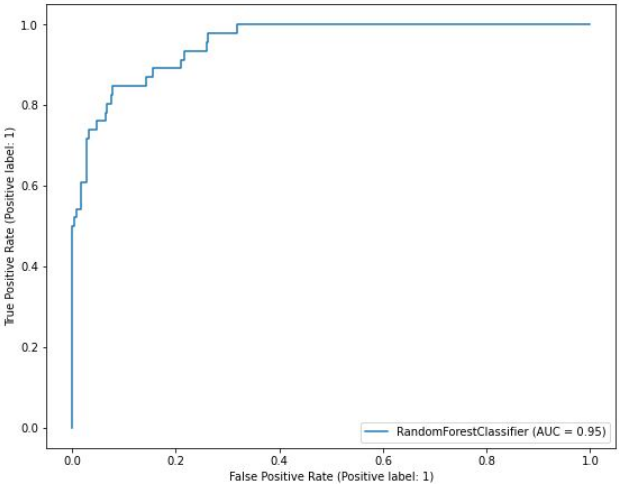


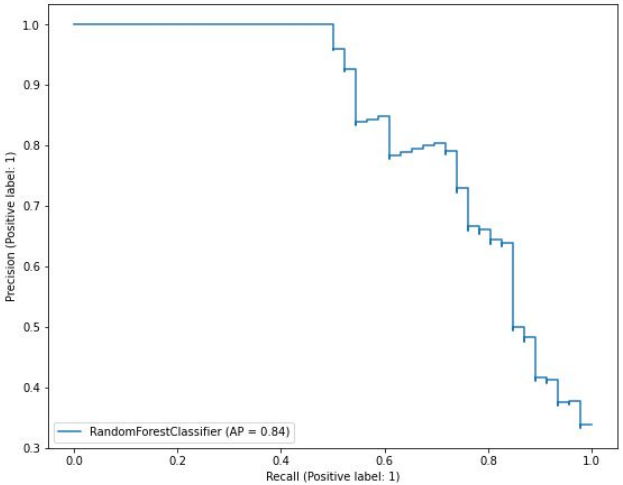
Hyperparameter tuning: Random Forest Classifier

- Parameters:
  - **n_estimators**: The number of trees in the forest;
  - **criterion**: The function to measure the quality of a split;
  - **max_depth**: The maximum depth of the tree;
  - **max_features**: The number of features to consider when looking for the best split.
- Best parameters: {**criterion**: *entropy*; **max_depth**: *4*; **max_features**: *sqrt*; **n_estimators**: *200*}
- Although we get better *local* scores with higher tree depths and number of estimators, we limited this to lower numbers. This was done to prevent overfitting.
- By using the default values, the model tends to overfit since the nodes are expanded until all leaves are pure.
- This indicates that our model might be too complex.


Count
mean_test_score
0
5
10
15
20
0.71
0.72
0.73
0.74

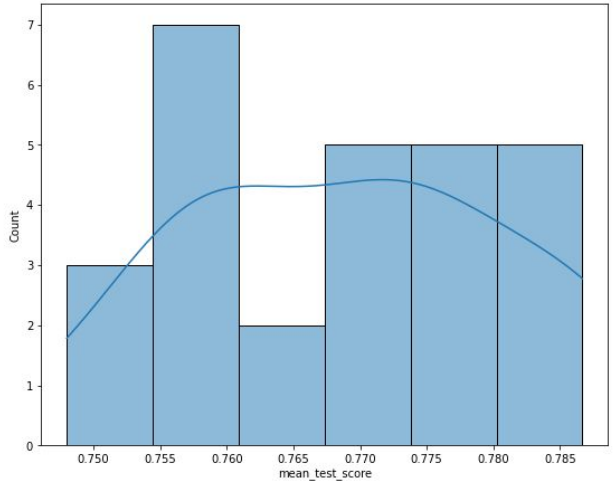

0.71
0.72
0.73
0.74
Mean test score (boxplot)

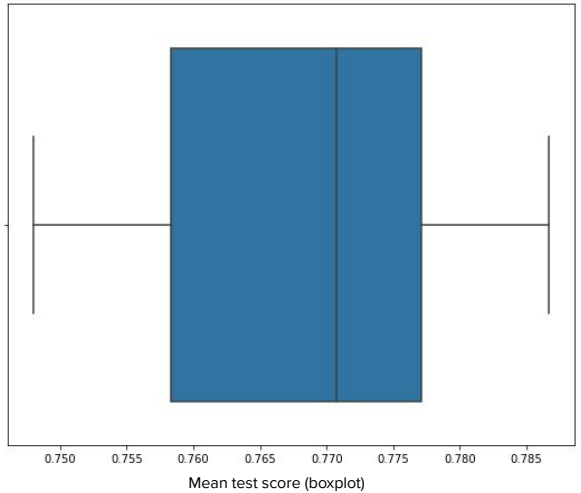

True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
RandomForestClassifier (AUC = 0.95)
0.0
0.2
0.4
0.6
0.8
1.0

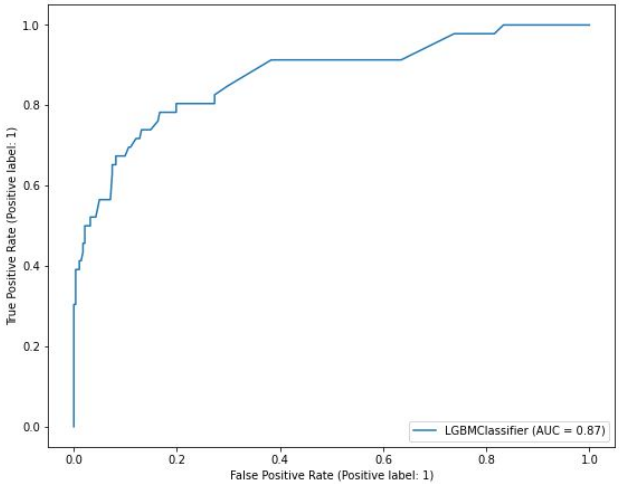

Precision (Positive label: 1)
Recall (Positive label: 1)
RandomForestClassifier (AP = 0.84)
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
0.0
0.2
0.4
0.6
0.8
1.0

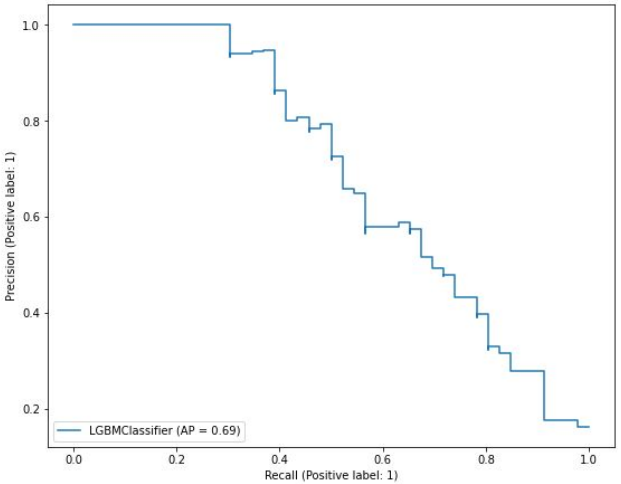
Hyperparameter tuning: LightGBM

- Parameters:
    - **max_depth**: Max depth for tree model;
    - **n_estimators**: Number of boosting iterations;
    - **num_leaves**: Max number of leaves in one tree.
- Best parameters: {**max_depth**: *1*, **n_estimators**: *50*, **num_leaves**: *2*}
- The **LightGBM** model prefers low max depths and low number of estimators. This suggests that the model is too complex.
- Similarly to the Random Forest Classifier, the results obtained suggest that our model is complex and could use more fine-tuning (maybe on the feature selection part).

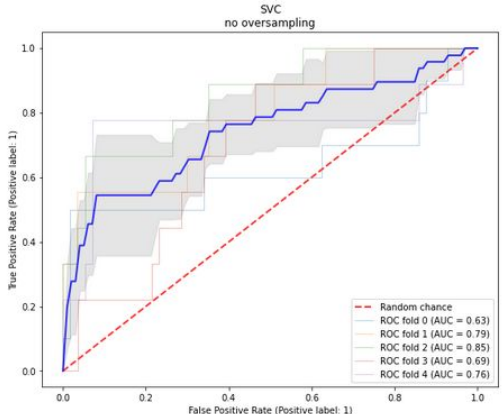

Count
mean_test_score

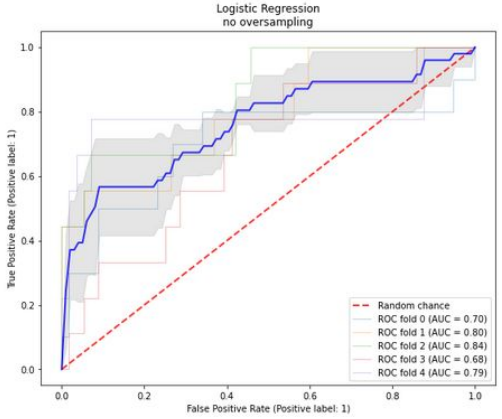

Mean test score (boxplot)

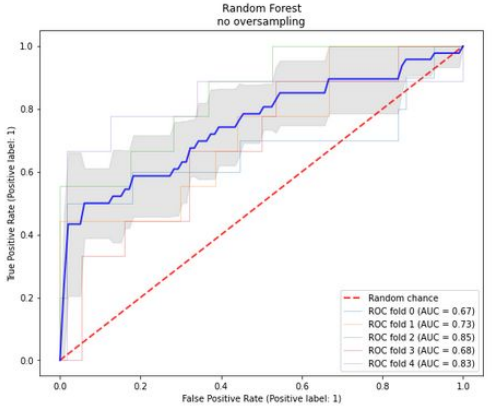

True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
LGBMClassifier (AUC = 0.87)

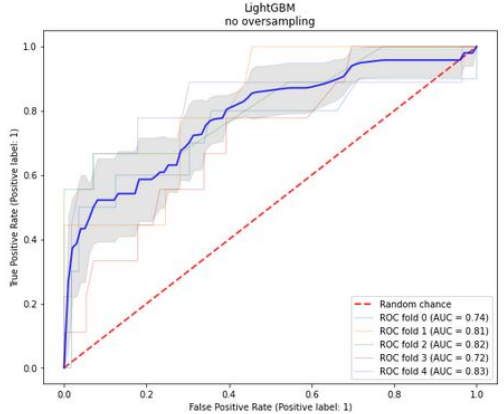

Precision (Positive label: 1)
Recall (Positive label: 1)
LGBMClassifier (AP = 0.69)

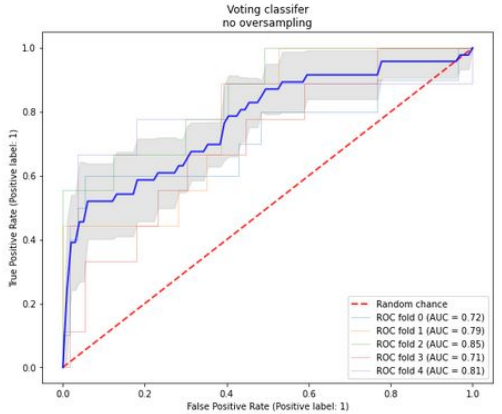
AUC - No Oversampling











Result discussion

- It was expected that the **SVC** wouldn't be able to get *good* results: there aren't many obvious boundary regions to separate accepted and rejected loans.
- At first, it was expected that the Linear kernel would yield the best results for the **SVC**, but the best results were obtained by the *Poly* kernel. This is probably related to the results observed for the **Random Forest** and the **LightGBM** classifiers: the model is too complex. The *Gamma* parameter in the *Poly* kernel helps the model regulate itself.
- Both the **Random Forest** and the **LightGBM** classifier obtain the best results. This success is caused by the existence of attributes that are much more important than others (e.g.: *balance_below_5k*, *balance_growth_lastsemester*) allowing the trees to narrow down on the result quickly.
- Given that our model was likely too complex, these tree-based algorithms tended to overfit on the training data.
- Since the **SVC** didn't choose the **Sigmoid** curve, it wasn't expected that the **Logistic Regression** would obtain good results.
- Our problem isn't *linear*. Tree-based methods are a better fit for this kind of problem.

Result discussion — continuation

- The over-sampling techniques used didn’t improve the results.
- Although not significant, SMOTE improved the results in a few select cases.
- ADASYN improved the local results by a lot, but it got the worst results in the Kaggle competition.
- Hyper-parameter tuning improved the results significantly, specially in the case of the **Support Vector** classifier.
- With more computing resources available, the usage of wrapper methods for feature selection should be attempted.
- The usage of different models for embedded method feature selection should be attempted: only the **Random Forest** and the **LightGBM** classifiers were used for this.
- Although it was decided not feasible due to the small amount of entries in the dataset, the combination of under-sampling techniques with the over-sampling ones could improve results.

Individual Factor

- Ana Inês Oliveira de Barros — **1.0**
- João de Jesus Costa — **1.0**
- João Lucas Silva Martins — **1.0**

References

- https://imbalanced-learn.org/stable
- https://lightgbm.readthedocs.io/en/latest/index.html
- https://matplotlib.org
- https://numpy.org
- https://pandas.pydata.org
- https://scikit-learn.org/stable/index.html
- https://scipy.org
- https://seaborn.pydata.org
- https://docs.scipy.org/doc/scipy/reference/stats.html